VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2017 Jan Kratochvíl

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Kratochvíl**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: EDIacademy, s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Peter Chovanec, Ph.D.**

Konzultant bakalářské práce: Ing. Lukáš Domin

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017 .....

EDIacademy, s.r.o.

Frýdecká 333
737 01 Český Těšín

Vážený pán

Jan Kratochvíl
U stavu 314/28
Havířov – Životice 736 01

Souhlas se zveřejněním bakalářské práce

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Českém Těšíně dne26. dubna 2017

Ing. Rostislav Zabystrzan
jednatel

Rád bych tímto poděkoval společnosti EDIacademy s.r.o. za tuto příležitost a své rodině za podporu při studiu.

Abstrakt

Tato bakalářská práce popisuje mou praxi u společnosti EDIacademy, s.r.o. Společnost se zabývá tvorbou výukových aplikací pro žáky základních škol. Mým hlavním úkolem bylo vytvoření aplikace Zvířátka dědy Lesoně, která slouží k výuce matematiky. Práce popisuje technologie, které byly použity ve vývoji a řešení praktických problémů.

Klíčová slova: HTML5, CSS, JavaScript, TypeScript, Phaser framework, EDIacademy s.r.o.

Abstract

This bachelor thesis describes my work experience at EDIacademy, s.r.o. The company focuses on creation of interactive teaching applications intended for pupils of elementary school. My main task was creating an application called Zvířátka dědy Lesoně, that assists teaching of mathematics. Contents of this thesis describe the technology used in development of this application and solutions to implementation challenges.

Key Words: HTML5, CSS, JavaScript, TypeScript, Phaser framework, EDIacademy s.r.o.

Obsah

Seznam použitých zkratek a symbolů

AABB	–	Axis-Aligned Bounding Box
CSS	–	Cascading Style Sheets
HTML	–	HyperText Markup Language
JSON	–	JavaScript Object Notation
OOP	–	Object Oriented Programming
SVN	–	Apache Subversion

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Tato práce popisuje průběh mé praxe u společnosti EDIacademy, s.r.o., která se zabývá tvorbou výukových aplikací. Pro praxi jsem se rozhodl z toho důvodu, že jsem chtěl uplatnit své teoretické znalosti při vývoji reálného softwaru.

1.1 O společnosti EDIacademy, s.r.o.

EDIacademy, s.r.o. [1] je společnost, která byla založena v roce 2015. Zaměřuje se na vývoj didaktických materiálů pro výuku matematiky. Společnost spolupracuje s Ostravskou Univerzitou a Univerzitou Komenského v Bratislavě, kde za spolupráce s fakultními základními školami provádí výzkum a testování nově vznikajících elektronických pomůcek pro výuku.

Projekt si klade za cíl rozvíjení klíčových kompetencí žáků základních škol k řešení problémů v oblasti kritického a logického myšlení a matematiky. K rozvoji jedince a jeho schopností se používají moderní technologie, které žáka vedou k samostatnému řešení problémů, užívání logického myšlení, práci s chybou, ověřování správnosti řešení a dalším dovednostem, které na konci základního vzdělání dosáhne [2].

Pomůcky vyvíjené EDIacademy, s.r.o. využívají Hejného výukovou metodu [3] [4]. Hejného výuková metoda je netradiční způsob výuky matematiky. Metodu zavedl profesor Milan Hejný za účelem vytvoření experimentálních úloh, které pomáhají studentům lépe porozumět matematickým a logickým problémům. M. Hejný založil společnost H-mat, o.p.s., se kterou společnost EDIacademy, s.r.o. od srpna 2015 spolupracuje.

1.2 Zadané úkoly

Mým hlavním úkolem bylo vytvořit interaktivní výukovou aplikaci nazvanou Zvířátka dědy Lesoně. V době mého nástupu do praxe tato aplikace teprve vznikala. Aplikace je spustitelná ve webovém prohlížeči, požadavkem bylo využití moderních HTML5 technologií [6]. Tuto aplikaci budou využívat primárně žáci základních škol a jejich učitelé. Bylo tedy žádoucí, aby aplikace byla vizuálně zajímavá pro děti a zároveň intuitivně ovladatelná. Byl mi dodán malý vzorový projekt, na základě kterého jsem aplikaci Zvířátka dědy Lesoně dále rozvíjel.

V pozdější fázi vývoje jsem z této aplikace vytvořil základní aplikaci nazvanou EDICore, která se dá využít pro vývoj budoucích aplikací pro EDIacademy, s.r.o. Ve vývoji jsem byl zodpovědný za vytváření webové stránky a programování logiky aplikace. Obrázky pro grafiku a dodatečné zvuky mi byly dodány.

Původně byla aplikace Zvířátka dědy Lesoně mým jedinným projektem. Časová náročnost na vývoj této aplikace byla tehdy odhadnuta na 400 hodin. Vzhledem k dodatečné implementaci dalších systémů a pozdějšímu vývoji projektu EDICore se časová náročnost zvýšila a přesáhla zhruba 600 hodin.

2 Používané technologie

Mým prvním úkolem v práci bylo seznámení se s technologiemi, se kterými společnost již pracovala a má s nimi zkušenosti. Jedná se o následující technologie:

2.1 Apache Subversion

Zkráceně SVN [7]. Je to open-source systém pro správu a verzování zdrojových kódů. Princip spočívá v ukládání verzí do centrálního úložiště nazvané repozitář. Nahráním změn do repozitáře se vytvoří nová verze - takzvaný *commit*. Uživatel má možnost skrze tento systém sledovat změny mezi jednotlivými verzemi, tedy commity.

V praxi jsme SVN používali pro správu zdrojových kódů všech projektů. Pro projekty, na kterých jsem pracoval, byl tento systém postačující.

2.2 HTML, CSS3, JavaScript

HTML [8] je značkovací jazyk používaný pro tvorbu webových stránek. Využití HTML je nutností, jelikož se jedná o vývoj webové aplikace. Pro nastavení zobrazení elementů HTML stránky se používá CSS [9]. Funkcionalitu webových stránek zajišťuje skriptovací jazyk JavaScript [10].

Teoreticky může být veškerá struktura stránky, její vzhled i funkcionalita zahrnuta v jednom HTML souboru. Avšak je vhodné tyto úseky rozdělit do příslušných souborů. Je tak rozdělena struktura stránky, její vzhled a funkcionalita.

Při tvorbě webové stránky, na které se aplikace Zvířátka dědy Lesoně zobrazuje, a ovládacích prvků aplikace jsem využil znalostí získaných v předchozím studiu. Aktuálně je standardem HTML verze 5, známé jako HTML5. Více informací o HTML5 lze najít na stránkách w3schools [6] nebo publikaci *Cameron Dane - A Software Engineer Learns HTML5, JavaScript and jQuery* [5].

2.3 TypeScript

TypeScript [11] je jazyk vyvíjený společností Microsoft, který je nadstavbou jazyka JavaScript. Zavádí statické typování proměnných a koncepty OOP do beztypového prostředí jazyka JavaScript. Zdrojový kód se překládá do čistého JavaScriptu. Platí však, že kód JavaScriptu je zároveň kódem TypeScriptu. Z vlastní zkušenost můžu říct, že zlepšuje přehlednost v kódu, protože umožňuje rozdělení logických celků do tříd a využít polymorfismus. Má oficiální podporu v několika oblíbených vývojářských prostředích, například Visual Studio.

Při použití JavaScript knihoven se do projektu přidaly dodatečné soubory nazvané Definition Files dostupné z Definitely Typed [12]. Obsah těchto souborů říká překladači TypeScriptu, jakou má knihovna strukturu, třídy a datové typy. Dodává tedy vývojářským prostředím možnost inteligentní nápovědy při psaní kódu.

Snadný způsob, jak se seznámit s jazykem TypeScript, je oficiální návod od Microsoftu, nazvaný TypeScript Handbook [13]. Pro hlubší porozumění je vhodná publikace *Nathan Rozenthals - Mastering TypeScript* [14].

2.4 Framework Phaser

Framework Phaser [15] je JavaScript framework primárně určený k vývoji her pomocí HTML5 technologií. Je veřejně dostupný pod licencí MIT a obsahuje velmi podrobnou dokumentaci. Framework nabízí zobrazování obrázku a animací, přehrávání videa a zvuku. Verze 2.4.8.[16], se kterou jsem pracoval, nabízí tři základní systémy simulace fyziky:

1. Arcade - základní simulace fyziky.
2. Ninja - systém fyziky vhodný pro složitější platformí hry. Podporuje AABB kolize [17].
3. P2 [18] - systém fyziky, který zvládne simulovat komplexní kolize, tření, restituci a fyzické vazby. Při práci na projektech byl využit tento systém.

Na stránkách frameworku se nachází mnoho oficiálních návodů [19] a návodů vytvořených komunitou [20]. Aktuálně stránky nabízí téměř 700 příkladů využití se zdrojovým kódem [21].

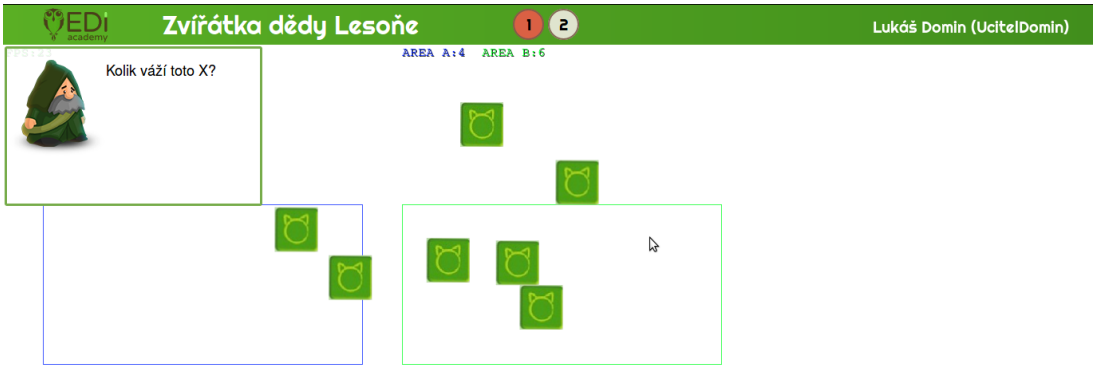
2.5 JavaScript knihovny

Ve vývoji aplikace bylo použito několika JavaScript knihoven:

1. Sortable.js [22] vytváří ovládací prvky, které slouží k přeskupení záznamů v listu. V aplikaci je knihovna využita pro přeskupení úloh.
2. Knihovna Spin.js [23] vytvoří animovaný Spinner. Označuje probíhající práci na pozadí. Graficky je zobrazen jako otáčivé kolo.
3. V několika případech je použita knihovna jQuery [24]. Konkrétně je využita při komunikaci se serverem a v několika krajních případech, kdy se funkcionalita JavaScriptu liší napříč různými prohlížeči.

2.6 Počáteční projekt

Na začátku praxe mi byl dodaný malý vzorový projekt, který mi pomohl se lépe zorientovat ve frameworku Phaser a jazyce TypeScript. Projekt obsahoval třídy pro přepínání mezi úlohami a základní komunikaci mezi serverem. Dále spouštěl framework Phaser a jeho zabudovaný systém simulace fyziky nazvaný P2. V poslední řadě obsahoval příklad vytvoření nového Sprite objektu - žetonu - se zapnutou klouzací a kolizní fyzikou. Na obrázku 1 je zobrazena plocha aplikace v počátečním projektu.



Obrázek 1: Počáteční projekt

3 Projekt Zvířátka dědy Lesoně

Jedná se o interaktivní 2D aplikaci určenou primárně dětem prvního stupně základní školy. Cílem aplikace je pomoci porozumět matematickým rovnicím a nerovnicím formou interaktivní výuky. Aplikaci také využívají učitelé, kteří tvoří zadání pro své žáky.

Žáci se přihlásí do webového portálu EDIacademy, kde vidí zadání úloh od svých učitelů. Po výběru zadání se aplikace spustí. Ihned po spuštění se načte balíček úloh pro daného žáka a spustí se první úloha.

Aplikace je ohraničena horní a spodní lištou, na kterých se nachází ovládací prvky. V horní liště se nachází název aplikace, přepínač mezi úlohami nazvaný *ProgressBar* a jméno uživatele. Ve spodní liště je tlačítko *Zpět*, které slouží pro vrácení poslední provedené změny, a tlačítko *Hotovo*, po jehož stisku se provede kontrola řešení. V levém horním rohu se nachází textové zadání úlohy.

Každá úloha obsahuje jednu až dvě rovnice či nerovnice, které je potřeba správně vyřešit. Rovnice obsahují pole, která představují strany rovnic. Mezi těmito poli se vyskytují znaménka *větší než*, *menší než*, *rovná se* nebo znaménko není známé. V tom případě je zobrazen *otazník*. Cílem žáka je správně vyřešit úlohu tak, aby platila matematická pravidla. Řešení není omezeno časem ani počtem pokusů.

3.1 Žetony

Hlavním prvkem hracího pole jsou žetony. Graficky jsou zobrazeny jako ikonky připomínající zvířátka. Tabulka 1 zobrazuje jednotlivé žetony, zvířátka, která představují a jejich číselné hodnoty. Děti mají na základě Hejného výukové metody asociována jednotlivá zvířátka s jejich číselnou hodnotou - váhou. Vedle žetonů se dále vyskytují dva speciální typy, které představují neznámé X a Y, nazvané masky. Na pořadí žetonů nezáleží, jelikož se žetony vždy sčítají.

Tabulka 1: Jednotlivé žetony, jejich grafická reprezentace a číselná hodnota

Grafika	Zvířátko	Hodnota	Grafika	Zvířátko	Hodnota
	Myš	1		Beran	6
	Kočka	2		Kráva	10
	Husa	3		Vůl	20
	Pes	4		Proměnná X	-
	Koza	5		Proměnná Y	-

Pokud jsou v zadání úlohy masky, musí mít vždy určenou hodnotu pro vyřešení úlohy. Při změně hodnoty v masce se nastaví všechny masky odkazující na stejnou proměnnou.

Po načtení úlohy se žetony mohou nacházet v několika polích:

1. V polích rovnic. V zadání úlohy jsou dány hodnoty, které se v rovnici již vyskytují.
2. Žák může vytáhnout žetony z příslušných zásobníků. Jednotlivé zásobníky mohou být povoleny nebo zakázány v zadání úlohy. Počet žetonů, které lze ze zásobníku vytáhnout, není omezen. Zásobníky se nachází pod vrchní lištou a graficky jsou zobrazeny jako žetony v oranžové barvě. Pokud je uživatel v módu Generátor úloh 4.6, u zásobníků jsou zobrazeny přepínače, pomocí kterých se jednotlivé zásobníky povolí nebo zakážou.
3. Bank s danými hodnotami. Vyprázdnění banku není podmínkou úspěšného řešení. Nachází se pod zásobníky, žetony zde vytvořené jsou určené v zadání úlohy.

3.2 Herní módy

Každé úloze přísluší vždy jeden herní mód:

1. Jeden řádek - jednoduchá rovnice se dvěma stranami.
2. Dva řádky - představuje soustavu dvou rovnic. Pokud se vyskytují masky, vzniká situace soustavy dvou rovnic o jedné či dvou neznámých.
3. Dva týmy - obdoba módu Dva řádky, pouze v tomto případě jsou pole větší. Tato verze je primárně určena pro volbu znaménka mezi stranami rovnic, avšak není to podmínkou.
4. Tři týmy - tento mód představuje libovolnou nerovnici se třemi stranami, mezi kterými mohou být libovolná znaménka.

Mezi rovnicemi se nacházejí znaménka. Podle typu znaménka se určuje způsob řešení úlohy:

1. Znaménko mezi stranami rovnice je pevně dané. Úloha se řeší přemístěním žetonů na strany rovnic tak, aby vždy platila matematická pravidla pro danou rovnici, případně nastavením hodnot v maskách.
2. Znaménko není určeno, místo něj se zobrazuje otazník. V této situaci je úkolem řešitele nastavit správné znaménko mezi stranami rovnic. Součty hodnot na stranách rovnic se nesmí změnit, student však může zaměnit například žeton *Kočka (2)* za dva žetony *Myš (1)*, aby si zjednodušil řešení. Znaménko se nastavuje poklepáním na jeho ikonku.

Správnost úloh se testuje pouze při stisku tlačítka *Hotovo*. Pokud je úloha vyřešena správně, zobrazí se vyskakovací okno oznamující správně vyřešení úlohy. Na obrázku 2 je zobrazeno prostředí aplikace po načtení úlohy.

Obrázek 2: Prostředí aplikace po načtení úlohy

3.3 Nápověda

V pravém horním rohu hrací plochy se nachází nápověda. Slouží jako vizuální pomůcka, která zobrazuje základní vztahy mezi hodnotami žetonů. Například *Pes (4)* je zobrazen jako součet *Husa (3) + Myš (1)*. Položky nápovědy, které se v dané úloze zobrazí, lze nastavit v zadání úlohy. Na obrázku 3 lze vidět nastavení viditelnosti položek nápovědy. Tyto přepínače jsou dostupné pouze v módu Generátoru úloh 4.6.

Obrázek 3: Nastavení nápovědy

3.4 Systém pravidel

Základní specifikace aplikace obsahovala pouze jednoduchá vstupní pole, do kterých měl zadavatel úlohy vepsat hodnoty žetonů očekávané v řešení úlohy. Osobně se mi tento přístup zdál omezený. Proto jsem navrhnul a naimplementoval sofistikovanější systém pravidel.

V nastavení úlohy lze pro každou stranu rovnice zadat pravidla, která se berou v potaz při vyhodnocení správnosti řešení. Nastavení pravidel je zobrazeno na obrázku 4. Položky pravidel jsou následující:

1. Obsahuje přesně - žetony v poli musí přesně odpovídat zadaným hodnotám v tomto pravidle. Pravidlo má exkluzivní prioritu; pokud je vyplněno, ostatní pravidla jsou ignorována. Vstupem jsou čísla oddělená čárkami.

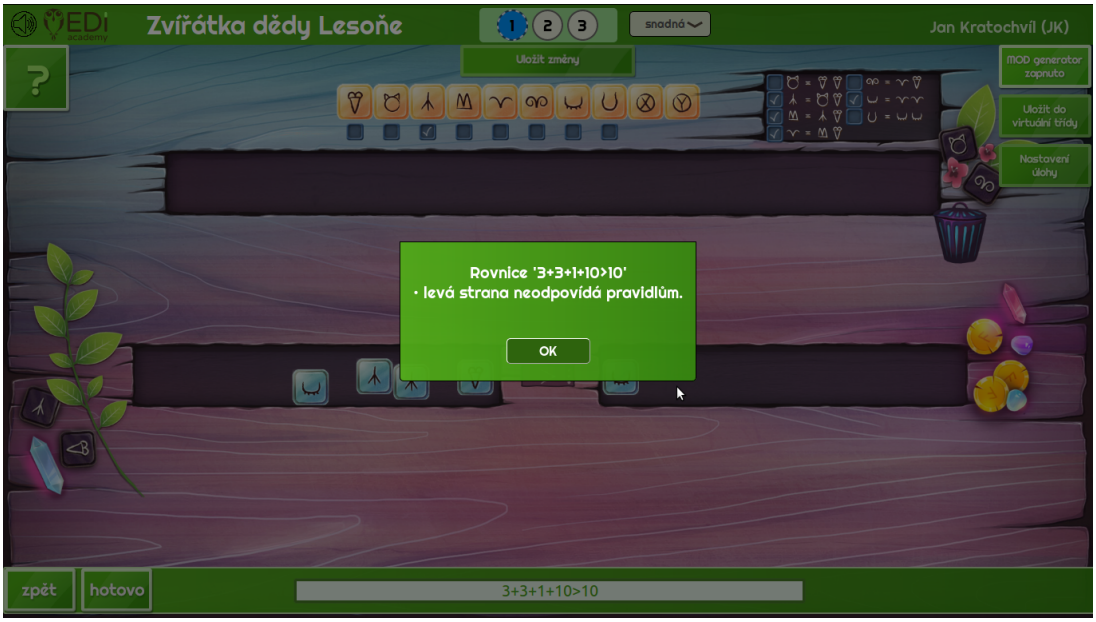
2. Součet pole je - vstupem je očekávaný součet hodnot žetonů a masek v poli.

3. Obsahuje pouze - definuje povolené hodnoty žetonů v poli. Vstupem jsou čísla oddělená čárkami. Zobrazí se chyba, pokud se zde vyskytuje hodnota z pole *Nesmí obsahovat*.

4. Nesmí obsahovat - zakázané hodnoty. Nesmí obsahovat hodnoty z polí *Obsahuje pouze* a *Obsahuje alespoň jednou*.

5. Obsahuje alespoň jednou - každá hodnota žetonu se v poli musí vyskytovat minimálně jednou. Vstupem jsou hodnoty oddělené čárkami. Je hlášena chyba, pokud se zde vyskytuje hodnota z pole *Nesmí obsahovat*.

Obrázek 4: Nastavení pravidel a zobrazení chyby.



Obrázek 5: Oznámení o porušení pravidel při řešení úlohy.

Pokud je uživatel v módu Generátor úloh 4.6, je zobrazen podrobný výpis, kde se nachází chyba v řešení nebo které pravidlo je porušeno. Tento výpis je znázorněn na obrázku 5. V opačném případě je pouze oznámeno, zda je řešení správné nebo chybné.

3.5 Příklad řešení úlohy

Na obrázku 6 je snímek ukázkové úlohy. Bank je prázdný a zásobníky nejsou dostupné. Řešitel tedy pracuje pouze se žetony v rovnicích. Všechny položky v nápovědě jsou dostupné. Úloha je typu *3 týmy*, s tím, že znaménko mezi první a druhou stranou je pevně dané a druhé znaménko je neurčité. V polích se objevuje dvakrát maska X.

Cílem řešení úlohy je najít hodnotu, kterou lze dosadit za masku X tak, aby platila rovnost mezi součtem první a druhé strany a zároveň určit vztah mezi druhou a třetí stranou. Nastavení masky je zobrazeno na obrázku 7.

Správným řešením úlohy, které je zobrazeno na obrázku 8, je dosadit za masku X zvířátko *Kočka (2)*. Součet hodnot v prostřední straně tedy činí 12, součet ve třetí straně je 27. Znaménko mezi druhou a třetí stranou se nastaví na *menší než* a úloha je správně vyřešena.

Obrázek 6: Zadání ukázkové úlohy.

Obrázek 7: Volba hodnoty v masce.

Obrázek 8: Správné řešení úlohy.

4 Projekt EDICore

Vzorový projekt, se kterým jsem začínal mou praxi, vznikl úpravou podobné, již existující aplikace od EDIacademy, s.r.o. Původní aplikace využívala stejné technologie, které byly použity při vývoji aplikace Zvířátka dědy Lesoně. Původní aplikace a nově vytvořená aplikace Zvířátka dědy Lesoně tedy sdílí určité části kódu, které se od té doby vyvíjely nezávisle. Motivací pro vývoj projektu nazvaném EDICore bylo vytvoření základní aplikace, která zastřešuje chování, jenž je identické mezi oběma aplikacemi. Příkladem je načítání dat, práce s úlohami a komunikace se serverem. Znamená to, že každá další aplikace využívající EDICore již nemusí tyto funkcionality řešit individuálně.

Pro vývoj EDICore byla použita aplikace Zvířátka dědy Lesoně. Krok po kroku jsem vyčleňoval funkcionalitu, která není unikátní pro danou aplikaci, a vytvořil z toho nový projekt. Postupně se EDICore stal základ aplikace a Zvířátka dědy Lesoně tento základ používá a modifikuje pro konkrétní účely.

4.1 Přidání do projektu

EDICore byl nahrán na nový SVN repozitář, který se do nové aplikace připojí pomocí nastavení SVN vlastnosti *svn:externals*. Tohoto se docílí vytvořením souboru s názvem *svn.externals* a vložením záznamu 1.

```
EDICore path_to_online_repository
```

Výpis 1: Záznam v souboru svn.externals

Záznam říká, že do složky s názvem *EDICore* se zahrne SVN repozitář nalezený v zadané cestě. Ve složce nové aplikace se použije příkaz, který řekne SVN, aby použil externí repozitáře uvedené v souboru, v tomto případě pouze repozitář EDICore. Příkaz je popsán ve výpisu 2.

```
svn propset svn:externals -F svn.externals .
```

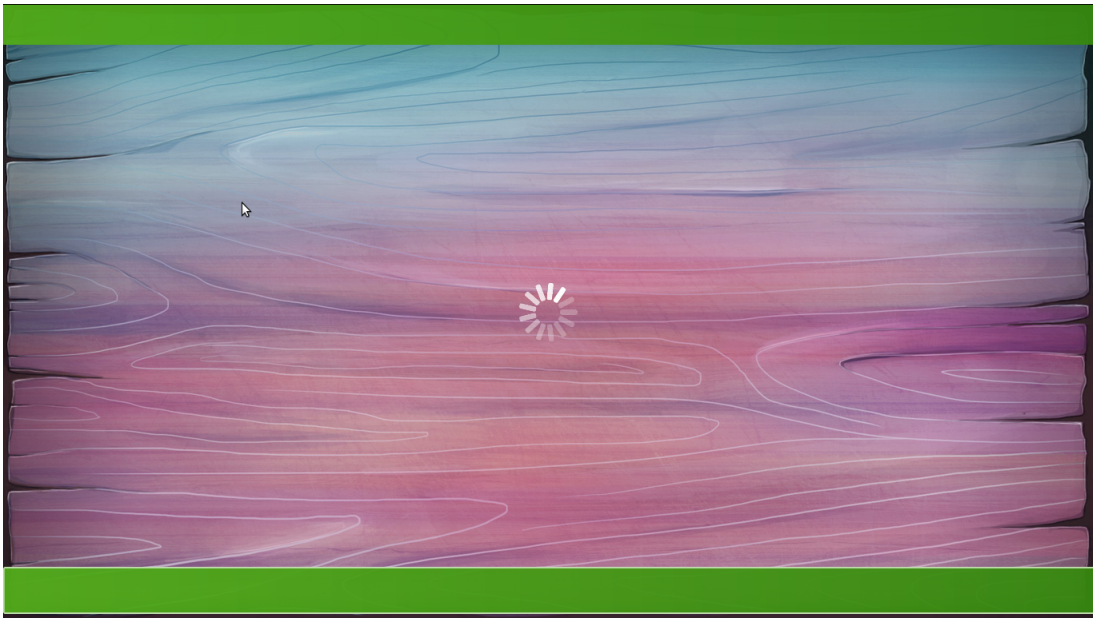
Výpis 2: Nastavení SVN:externals na repozitář

4.2 Průběh načítání aplikace

Čistý projekt EDICore po načtení ve webové stránce provede následující:

1. Inicializace frameworku Phaser v módu 2D vykreslování Canvas. Zobrazení animovaného Spinneru při načítání dat na pozadí, znázorněno na obrázku 9.

2. Fáze *Preloader* - načtení dat do paměti. Konkrétně se jedná o obrázky pro použití v hracím poli, zvuky ve formátu .ogg a testovací data ve formátu JSON, která se použijí při offline testování.



Obrázek 9: Zobrazen Spinner, zatímco se na pozadí načítá aplikace.

3. Spuštění *Environment* - hlavní stav prostředí samotné aplikace. V tomto stavu se vytváří pomocné objekty pro běh aplikace. Jedná se o manažer zvuku, manažer vyskakovacích oken, manažer úloh, ProgressBar a Generátor úloh. Na pozadí je kontaktován server s požadavky o stažení balíčku úloh a informací o uživateli ve formátu JSON. Pokud se nepodaří server kontaktovat a doména stránky se nachází na testovacích serverech, použijí se data pro offline testování, která byla načtena ve fázi Preloader. Aplikace zůstává v módu načítání do té doby, než se podaří získat data o uživateli a úlohách. Poté se vytvoří ovládací prvky pro přepínání úloh v *ProgressBaru*, který se nachází v zelené liště na horním okraji stránky.

4. Aplikace je připravena k provozu, načítání je dokončeno a animovaný Spinner se odstraní. V případě aplikace Zvířátka dědy Lesoně se nespouští žádné hlavní menu, místo toho je načtena první úloha a aplikace je v provozu.

4.3 Využití EDICore

Jak již bylo zmíněno, aplikace Zvířátka dědy Lesoně je modifikací základní aplikace EDICore. Systém načítání podporuje záměnu původní třídy za vlastní. Přestože jazyk TypeScript podporuje OOP třídy, kód se překládá do JavaScriptu, kde žádné třídy neexistují. Místo toho JavaScript používá prototypy, které se při vytváření nového objektu klonují.

Byla vytvořena třída *CoreClasses*, která uchovává odkazy na prototypy objektů. Tyto odkazy lze přepsat odkazy na prototypy vlastních tříd. Tento způsob má však nevýhodu, protože prototypy objektů se ukládají do proměnných typu *any*. Proměnná typu *any* znamená *jakýkoliv datový typ*. Při vytváření nového objektu ztrácí vývojářské prostředí informace o typu objektu,

nezobrazuje nápovědu a při nepozornosti vývojáře může dojít k chybám za běhu. Ve výpise 3 lze vidět kód, který spouští celou aplikaci.

```
namespace EDICore {
  /**
  * Class with various static variables
  * that should point to class prototypes
  */
  export class CoreClasses {
      public static Ajaxer: any;
      public static App: any;
      public static Env: any;
      public static Generator: any;
      public static Preloader: any;
      public static ProgressBar: any;
      public static SnapshotManager: any;
      public static Task: any;
  }

  /**
  * Declare a function that overrides default values in CoreClasses
  */
  export let assignCustomCoreClasses: Function;

  /**
  * Set up the default prototypes
  */
  function assignDefaultCoreClasses() {
      EDICore.CoreClasses.App = EDICore.EdiApp;
      EDICore.CoreClasses.Ajaxer = EDICore.Ajaxer;
      EDICore.CoreClasses.Env = EDICore.Env;
      EDICore.CoreClasses.Generator = EDICore.Generator;
      EDICore.CoreClasses.Preloader = EDICore.Preloader;
      EDICore.CoreClasses.ProgressBar = EDICore.ProgressBar;
      EDICore.CoreClasses.SnapshotManager = EDICore.SnapshotManager;
      EDICore.CoreClasses.Task = EDICore.Task;
  }

  window.onload = () => {
      // assign default class prototypes into CoreClasses
```

```
        assignDefaultCoreClasses();
        if (assignCustomCoreClasses) {
            // if the override function is defined,
            // assign custom class prototypes into CoreClasse
            assignCustomCoreClasses();
        }

        // start up main App
        const canvas = document.getElementById("content");
        const edi = new EDICore.CoreClasses.App(canvas);
    };
}
```

Výpis 3: Zjednodušený výpis ze souboru CoreClasses.ts

Po načtení stránky se ve funkci *assignDefaultCoreClasses()* přiřadí původní prototypy do proměnných ve třídě *CoreClasses*. Poté se zavolá funkce *assignCustomCoreClasses()*, ve které se očekává přepsání odkazů na prototypy vlastních podtříd. Příklad využití vlastních podtříd v aplikaci Zvířátka dědy Lesoně ve výpisu 4:

```
// here point towards custom class prototypes
EDICore.assignCustomCoreClasses = () => {
    EDICore.CoreClasses.App                 = Leson.LesonApp;
    EDICore.CoreClasses.Env                 = Leson.Env;
    EDICore.CoreClasses.Generator           = Leson.Generator;
    EDICore.CoreClasses.Preloader           = Leson.Preloader;
    EDICore.CoreClasses.Task                = Leson.LesonTask;
};
```

Výpis 4: Zjednodušený výpis ze souboru LesonClasses.ts

V kódu základní aplikace EDICore se všechny objekty vytváří s využitím statických proměnných ve třídě *CoreClasses*, příklad ve výpisu 5:

```
this.soundHelper = new EDICore.CoreClasses.SoundHelper(this);
this.ajaxer = new EDICore.CoreClasses.Ajaxer(this, onDataLoaded);
```

Výpis 5: Využití prototypů z třídy CoreClasses

4.4 Vyskakovací okna

Další důležitou částí projektu EDICore bylo vytvoření systému vyskakovacích oken, která slouží pro komunikaci s uživatelem. Okna nahradila funkce zabudované do prohlížečů *alert()* a *confirm()*. Okna obsahují callbacky - zpětně volané funkce - které se volají při zavření oken.

1. Prvním krokem bylo vytvoření základního okna, které se při zobrazení animuje efektem fade-in. Třída byla nazvána *BasePopup*.

2. Z této třídy byla vytvořena podtřída ModalPopup. Třida zobrazuje dotaz uživateli a nabízí dvě volby - tlačítka s nastavitelným textem. Okno nahradilo zabudovanou funkci prohlížeče *confirm()*. Zobrazeno na obrázku 10.

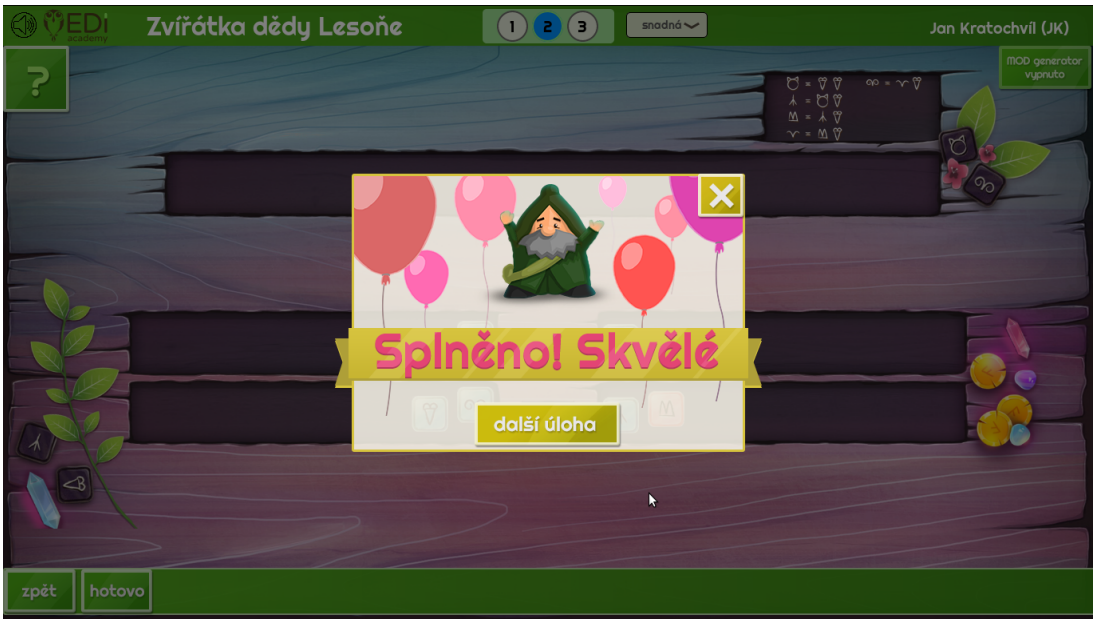
3. Zobrazení informací nebo oznámení - *AlertPopup*. Obsahuje pouze jedno tlačítko pro zavření okna. Zobrazeno na obrázku 11.

4. Okno *SuccessPopup*. Zobrazí se při správném řešení úlohy po stisku tlačítka *Hotovo*. Obsahuje dvě tlačítka - zavření okna k návratu do aplikace a tlačítko pro přesun na další úlohu. Zobrazeno na obrázku 12.

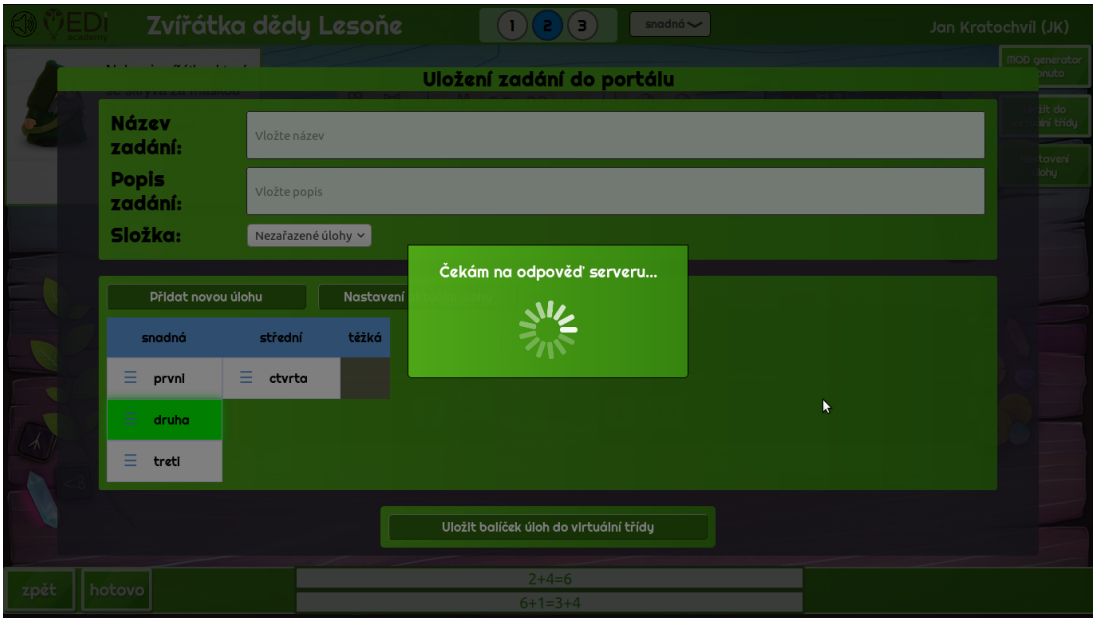
5. *WaitPopup* - toto okno zablokuje aplikaci a zobrazí text, zatímco aplikace čeká na dokončení nějaké práce. Okno obsahuje text a animovaný Spinner. Je primárně použito při odesílání dat na server a čekání na odpověď. Okno může být zobrazeno s časovým limitem - timeout - po jehož vypršení se okno zavře. Zobrazeno na obrázku 13.

Obrázek 10: Dotaz na uživatele, zda si přeje restartovat úlohu.

Obrázek 11: Uživatel se pokusil o nesprávné řešení úlohy.



Obrázek 12: Úloha byla správně vyřešena.



Obrázek 13: Aplikace čeká na odpověď ze serveru.

Nastal však problém při zobrazování několika oken za sebou, protože se zobrazovala přes sebe. Bylo nutné implementovat systém fronty pro vyskakovací okna. Vznikla třída *QueuedPopup*, která se stala základní třídou pro výše zmíněné třídy. Předtím, než se okno zobrazí, třída provede kontrolu, zda se právě nezobrazuje jiné okno. Pokud ano, nové okno se přidá do fronty zobrazení, v opačném případě se okno ihned zobrazí. Po zavření okna a dokončení jeho animace se kontroluje obsah fronty oken. Jestliže fronta obsahuje další záznam, zavolá se metoda pro zobrazení dalšího okna.

Každé okno je tvořeno dvěma hlavními HTML prvky. První je *div* element nazvaný *overlay*, který překryje celou plochu aplikace s průhledným šedým pozadím. Prvek má nastavený způsob zobrazení na *flex* [25]. Zaručuje vycentrování dceřiných prvků na obrazovce jak horizontálně, tak vertikálně. Prvek zamezuje interakci s aplikací na pozadí, zatímco se okno zobrazuje. Druhou částí je *div* element nazvaný *content* se samotným obsahem okna.

4.5 TaskManager

V počátku aplikace se úlohy přepínaly přes jednotlivá tlačítka úloh na vrchní liště. Později byly vytvořeny komponenty, které také načítaly úlohu a upravovaly ji. Každá změna v úloze se ihned uložila a chyběla možnost vracení změn. Proto jsem vytvořil jednotný řetězec odpovědnosti a zároveň systém pracovní verze úlohy. Tímto vznikla třída *TaskManager*, která má obecně na starost práci s úlohami:

1. Vytvoření pracovní kopie. Načtená úloha se duplikuje do proměnné *workingCopy* - tedy pracovní kopie - a až tato kopie se předává ostatním komponentám aplikace. Takto se

nemůže stát, že by se načtená úloha omylem změnila. Třída obsahuje metody, které uloží provedené změny a vytvoří novou pracovní kopii. Uložení změn je možné pouze při zapnutém módu Generátoru úloh.

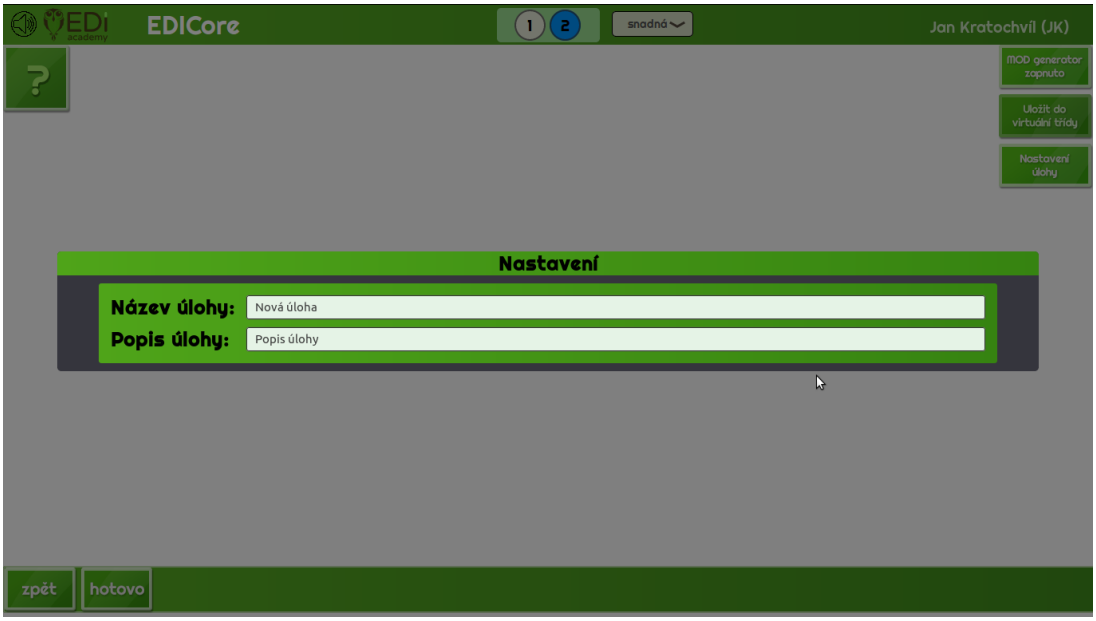
2. Přepínání úloh. Třída obsahuje metody, která provádí opatření k zajištění správnému chodu aplikace při změně úlohy. Před přepnutím na jinou úlohu se nejprve kontroluje, zda byla úloha změněna a případně zobrazí dotaz k uložení změn.

4.6 Generátor úloh

Ve specifikaci aplikace Zvířátka dědy Lesoně byl požadavek na systém, který dovoluje přepínat mezi módem řešení úlohy a módem úpravy úlohy, nazvaný Generátor úloh. Žáci nemají možnost se přepnout do módu Generování úloh. Tuto funkcionalitu mají dostupnou pouze učitelé a administrátoři. Pokud je mód Generátor úloh zapnutý, aplikace si hlídá jakékoli změny v úloze či jejím nastavení. Když se mód vypíná a byly provedeny změny, aplikace se dotáže uživatele, zda si přeje změny uložit či zahodit.

Přepnutí generátoru je ovládáno tlačítkem umístěným v pravém horním rohu aplikace. Po zapnutí módu Generátor se v aplikaci zobrazí dodatečné ovládací prvky k nastavení úlohy.

Tlačítko *Nastavení úlohy* zobrazí okno s možnostmi pro nastavení úlohy, zobrazené na obrázku 14. Zde zadavatel vkládá název úlohy a její popis, který se zobrazuje v levém horním rohu aplikace. Aplikace Zvířátka dědy Lesoně má vlastní úpravu tohoto okna, kde se nastavují povolené žetony v maskách a pravidla pro jednotlivé rovnice, zobrazené na obrázku 4 .

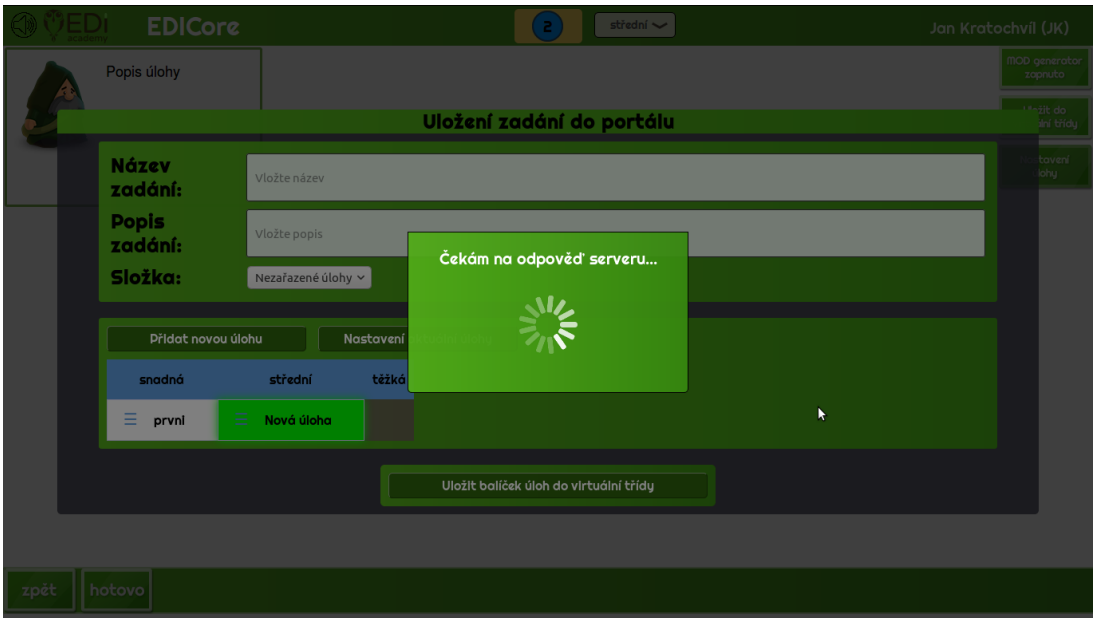


Obrázek 14: Nastavení úlohy v EDICore.

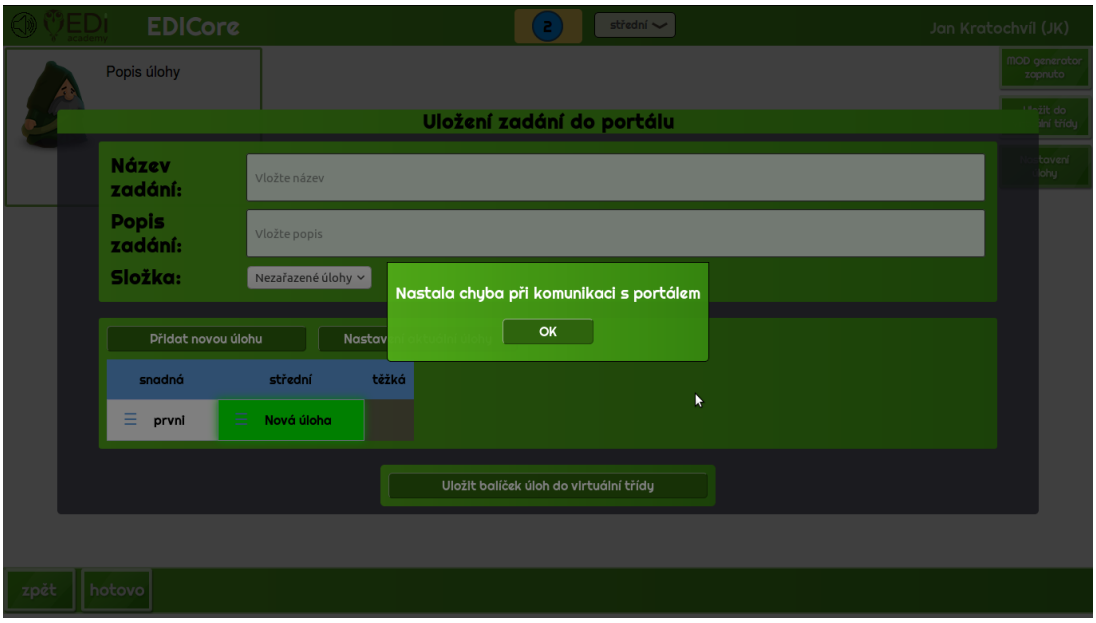
Obrázek 15: Okno Uložení zadání do portálu.

Na obrázku 15 je zobrazeno okno, které slouží k nastavení balíčku úloh, nazvané *Uložení zadání do portálu*. V tomto okně se nastaví název a popis aktuálního balíčku úloh. Dále se zde nachází správa úloh: mohou být přidány nové úlohy, přeskupeny za využití knihovny SortableJS či smazány. Na spodní hraně okna se nachází tlačítko pro uložení balíčku úloh na server. Při komunikaci je zobrazeno vyskakovací okno *WaitPopup*, viditelné na obrázku 16. Okno je zobrazeno do té doby, než aplikace obdrží odpověď ze serveru. Pokud komunikace selže, aplikace zobrazí uživateli chybové hlášení, znázorněno na obrázku 17.

Okna *Nastavení úlohy* a *Uložení zadání do portálu* využívají stejný systém vyskakovacích oken. Jsou zobrazovány s animací, avšak nejsou součástí fronty vyskakovacích oken. Mají nižší prioritu, tím pádem jsou tato okna překryta ostatními vyskakovacími okny.



Obrázek 16: Čekání na odpověď serveru.



Obrázek 17: Zobrazení chyby při komunikaci se serverem.

5 Závěr

V průběhu praxe jsem uplatnil teoretické znalosti, které jsem získal při studiu. Konkrétně se jedná o vývoj webové aplikace pomocí HTML5, CSS3 a JavaScript technologií. S jazykem TypeScript jsem se již v minulosti setkal. Díky zkušenostem s dalšími OOP jazyky, se kterými jsem se seznámil v průběhu studia, nebylo složité s jazykem TypeScript pracovat. Mé teoretické nedostatky jsem si doplnil z knižních publikací uvedených v literatuře.

Při práci na těchto projektech jsem dospěl k závěru, že je ideální průběžně vytvářet co nejpřehlednější dokumentaci a často ukládat změny do repozitáře SVN. Ke každému SVN commitu jsem připisoval výstižný komentář. V logu comittů chybělo formátování, které je potřeba k zachování přehlednosti. Proto jsem zavedl nový soubor, který fungoval jako *changelog*. Je to soubor obsahující detailní popis změn v jednotlivých verzích aplikace.

Výsledkem mé praxe je kompletní implementace webové aplikace Zvířátka dědy Lesoně. Z této aplikace vznikla základní aplikace s názvem EDICore, kterou lze použít k vývoji dalších aplikací pro společnost EDIacademy, s.r.o.

Působení v praxi hodnotím pozitivně a jsem velmi vděčný za tuto pracovní příležitost. Věřím, že znalosti a zkušenosti zde nabyté uplatním v budoucím studiu a později i v zaměstnání.

Literatura

[1] *EDIacademy, s.r.o.* [online]. [cit. 2017-04-17]. Dostupné z: http://www.ediacademy.cz/

[2] *Informace o projektu EDIacademy, s.r.o.* [online]. [cit. 2017-04-17]. Dostupné z: http://www.ediacademy.cz/#project

[3] *Co je to „Hejného metoda“* [online]. Praha, 2017 [cit. 2017-04-17]. Dostupné z: http://www.h-mat.cz/hejneho-metoda

[4] HEJNÝ, Milan, Jarmila NOVOTNÁ a Naďa STEHLÍKOVÁ. *Dvacet pět kapitol z didaktiky matematiky* [online]. 2. sv. Praha: Univerzita Karlova v Praze - Pedagogická fakulta, 2004 [cit. 2017-04-20]. ISBN 80-7290-189-3. Dostupné z: http://class.pedf.cuni.cz/NewSUMA/Download/Volne/SUMA_59.pdf

[5] CAMERON, Dane. *A Software Engineer Learns HTML5, JavaScript and jQuery* [online]. 1st Edition. CreateSpace Independent Publishing Platform, 2013 [cit. 2017-04-20]. ISBN 978-1493692613. Dostupné z: https://www.amazon.com/Software-Engineer-Learns-JavaScript-jQuery/dp/1493692615/

[6] *HTML5* [online]. [cit. 2017-04-18]. Dostupné z: https://www.w3schools.com/html/html5_intro.asp

[7] *Apache Subversion* [online]. [cit. 2017-04-17]. Dostupné z: https://subversion.apache.org/

[8] *HTML na w3schools* [online]. [cit. 2017-04-20]. Dostupné z: https://www.w3schools.com/html/

[9] *CSS3 na w3schools* [online]. [cit. 2017-04-20]. Dostupné z: https://www.w3schools.com/css/css3_intro.asp

[10] *JavaScript na w3schools* [online]. [cit. 2017-04-20]. Dostupné z: https://www.w3schools.com/js/

[11] *TypeScript* [online]. [cit. 2017-04-17]. Dostupné z: https://www.typescriptlang.org/

[12] *DefinitelyTyped* [online]. [cit. 2017-04-17]. Dostupné z: https://github.com/DefinitelyTyped

[13] *TypeScript Handbook* [online]. [cit. 2017-04-18]. Dostupné z: https://www.typescriptlang.org/docs/tutorial.html

[14] ROZENTHALS, Nathan. *Mastering TypeScript - Second Edition* [online]. 2nd Edition. Packt Publishing, 2017 [cit. 2017-04-20]. ISBN 978-1786468710. Dostupné z: https://www.amazon.com/Mastering-TypeScript-Second-Nathan-Rozentals-ebook/dp/B01DPR2EQC/

[15] *Phaser framework* [online]. [cit. 2017-04-17]. Dostupné z: http://phaser.io/

[16] *Dokumentace frameworku Phaser v2.4.8.* [online]. [cit. 2017-04-20]. Dostupné z: https://phaser.io/docs/2.4.8/index/

[17] *Detekce AABB kolizí ve 2D* [online]. [cit. 2017-04-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection

[18] *Systém fyziky P2* [online]. [cit. 2017-04-20]. Dostupné z: https://schteppe.github.io/p2.js/

[19] *Oficiální návody ke frameworku Phaser* [online]. [cit. 2017-04-20]. Dostupné z: https://phaser.io/learn/official-tutorials

[20] *Komunitní návody ke frameworku Phaser* [online]. [cit. 2017-04-17]. Dostupné z: https://phaser.io/learn/community-tutorials

[21] *Příklady využití frameworku Phaser* [online]. [cit. 2017-04-20]. Dostupné z: https://phaser.io/examples

[22] *SortableJS* [online]. [cit. 2017-04-17]. Dostupné z: https://github.com/SortableJS

[23] *SpinJS* [online]. [cit. 2017-04-17]. Dostupné z: http://spin.js.org/

[24] *jQuery* [online]. [cit. 2017-04-17]. Dostupné z: https://jquery.com/

[25] COYIER, Chris. *A Complete Guide To Flexbox* [online]. [cit. 2017-04-20]. Dostupné z: https://css-tricks.com/snippets/css/a-guide-to-flexbox/